bdiGDB

JTAG debug interface for GNU Debugger

XScale

User Manual

Manual Version 1.01 for BDI3000

abatron

©1997-2009 by Abatron AG

1 Introduction 3
1.1 BDI3000 3
1.2 BDI Configuration 4

2 Installation 5
2.1 Connecting the BDI3000 to Target 5
2.2 Connecting the BDI3000 to Power Supply 7
2.3 Status LED «MODE» 8
2.4 Connecting the BDI3000 to Host 9
2.4.1 Serial line communication 9
2.4.2 Ethernet communication 10
2.5 Installation of the Configuration Software 11
2.5.1 Configuration with a Linux / Unix host 12
2.5.2 Configuration with a Windows host 14
2.5.3 Configuration via Telnet / TFTP 16
2.6 Testing the BDI3000 to host connection 18
2.7 TFTP server for Windows 18

3 Using bdiGDB 19
3.1 Principle of operation 19
3.2 Configuration File 20
3.2.1 Part [INIT] 21
3.2.2 Part [TARGET] 23
3.2.3 Part [HOST] 27
3.2.4 Part [FLASH] 29
3.2.5 Part [REGS] 33
3.3 Debugging with GDB 35
3.3.1 Target setup 35
3.3.2 Connecting to the target 35
3.3.3 Breakpoint Handling 36
3.3.4 GDB monitor command 36
3.3.5 Debug handler 37
3.3.6 Target serial I/O via BDI 38
3.4 Telnet Interface 39
3.4.1 Command list 40
3.4.2 CPxx Registers 41
3.5 Multi-Core Support 42

4 Specifications 43

5 Environmental notice 44

6 Declaration of Conformity (CE) 44

7 Warranty 45

7 Appendices

A Troubleshooting 46

B Maintenance 47

C Trademarks 47

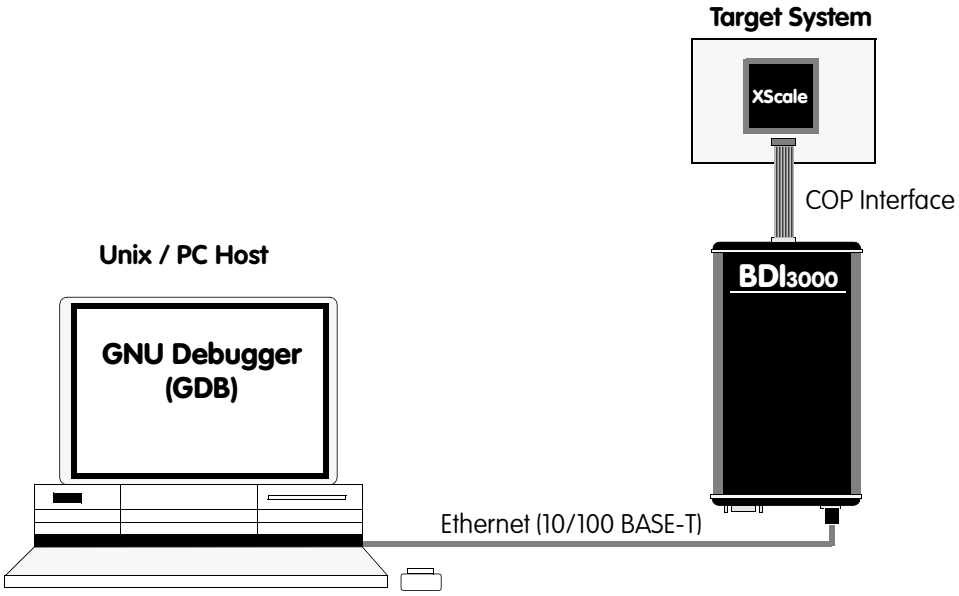
© Copyright 1997-2009 by ABATRON AG Switzerland

1 Introduction

bdiGDB enhances the GNU debugger (GDB), with JTAG debugging for XScale based targets. With the built-in Ethernet interface you get a very fast code download speed. No target communication channel (e.g. serial line) is wasted for debugging purposes. Even better, you can use fast Ethernet debugging with target systems without network capability. The host to BDI communication uses the standard GDB remote protocol.

An additional Telnet interface is available for special debug tasks (e.g. force a hardware reset, program flash memory).

The following figure shows how the BDI3000 interface is connected between the host and the target:



1.1 BDI3000

The BDI3000 is the main part of the bdiGDB system. This small box implements the interface between the JTAG pins of the target CPU and a 10/100Base-T Ethernet connector. The firmware of the BDI3000 can be updated by the user with a simple Linux/Windows configuration program or interactively via Telnet/TFTP. The BDI3000 supports 1.2 – 5.0 Volts target systems.

.

© Copyright 1997-2009 by ABATRON AG Switzerland

1.2 BDI Configuration

As an initial setup, the IP address of the BDI3000, the IP address of the host with the configuration file and the name of the configuration file is stored within the flash of the BDI3000.
Every time the BDI3000 is powered on, it reads the configuration file via TFTP.
Following an example of a typical configuration file:

```
; bdiGDB configuration file for IQ80310 board
; ----------------------------------------
;
[INIT]
WCP15   0x2001      0x00000001  ;Disable Write Buffer Coalescing
;
; Init DRAM
WM32    0x00001508  0xA0000000  ;SDBR: Set SDRAM Base Address
WM32    0x00001504  0x00000AA0  ;SDCR: 1 single-sided DIMM
WM32    0x0000150C  0x00000008  ;SBR0: 1 bank 32MB
WM32    0x00001510  0x00000008  ;SBR1: 1 bank 32MB
WM32    0x00001568  0x00000000  ;RFR : Diables Refresh Cycle
WM32    0x00001500  0x00000003  ;SDIR: Issue NOP cmd to SDRAM
DELAY   10                      ;delay after the NOP command
WM32    0x00001568  0x00000600  ;RFR : Set Refresh Rate
WM32    0x00001500  0x00000002  ;SDIR: Precharge all
WM32    0x00001500  0x00000004  ;SDIR: Auto Refresh #1
            ........
WM32    0x00001500  0x00000004  ;SDIR: Auto Refresh #8
WM32    0x00001500  0x00000000  ;SDIR: Send Mode Reg Set Cmd with CAS Latency 2
WM32    0x00001500  0x00000006  ;SDIR: Issue aNormal Operation command
;

[TARGET]
CPUTYPE         IOP310          ;the target CPU type
JTAGCLOCK       0               ;use 16 MHz JTAG clock
DBGHANDLER      0xFFFF0000      ;debug handler base address
ENDIAN          LITTLE          ;memory model (LITTLE | BIG)
BREAKMODE       SOFT            ;SOFT or HARD

[HOST]
IP              151.120.25.119
FILE            E:\cygwin\home\bdidemo\xscale\vxworks
FORMAT          BIN 0xA0020000
LOAD            MANUAL          ;load code code MANUAL or AUTO after reset

[FLASH]
WORKSPACE   0xa0020000  ;workspace in target RAM for fast programming algorithm
CHIPTYPE    STRATAX8    ;Flash type
CHIPSIZE    0x800000    ;The size of one flash chip in bytes
BUSWIDTH    8           ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE        E:\cygwin\home\bdidemo\xscale\vxworks
FORMAT      BIN 0x00080000
ERASE       0x00080000  ;erase sector 4
ERASE       0x000A0000  ;erase sector 5
ERASE       0x000C0000  ;erase sector 6
ERASE       0x000E0000  ;erase sector 7

[REGS]
FILE          E:\cygwin\home\bdidemo\xscale\reg80310.def
```

Based on the information in the configuration file, the target is automatically initialized after every reset.

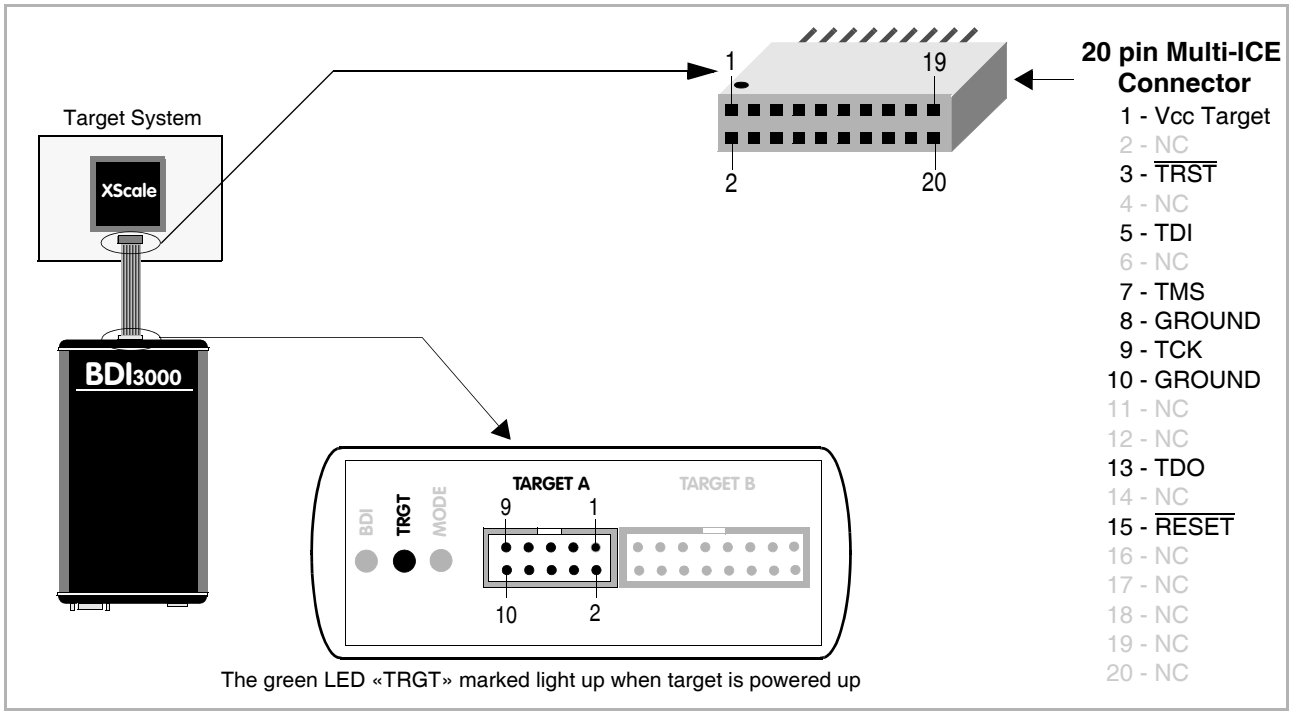
© Copyright 1997-2009 by ABATRON AG Switzerland

2 Installation

2.1 Connecting the BDI3000 to Target

The enclosed target cable is designed for the Intel recommended 20pin JTAG connector. In case where the target system has an appropriate connector, the cable can be directly connected. The pin assignment is in accordance with the Intel specification.

In order to ensure reliable operation of the BDI (EMC, runtimes, etc.) the target cable length must not exceed 20 cm (8").



The green LED «TRGT» marked light up when target is powered up

For BDI TARGET A connector signals see table on next page.

Warning:
Before you can use the BDI3000 with an other target processor type (e.g. PPC <--> ARM), a new setup has to be done (see chapter 2.5). During this process the target cable must be disconnected from the target system.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming a new firmware for an other target CPU.

© Copyright 1997-2009 by ABATRON AG Switzerland

TARGET A Connector Signals

Pin	Name	Description
1	reserved	This pin is currently not used.
2	$\overline{\text{TRST}}$	**JTAG Test Reset** This open-drain / push-pull output of the BDI3000 resets the JTAG TAP controller on the target. Default driver type is push-pull.
3+5	GND	**System Ground**
4	TCK	**JTAG Test Clock** This output of the BDI3000 connects to the target TCK line.
6	TMS	**JTAG Test Mode Select** This output of the BDI3000 connects to the target TMS line.
7	$\overline{\text{RESET}}$	This open collector output of the BDI3000 is used to reset the target system.
8	TDI	**JTAG Test Data In** This output of the BDI3000 connects to the target TDI line.
9	Vcc Target	**1.2 – 5.0V:** This is the target reference voltage. It indicates that the target has power and it is also used to create the logic-level reference for the input comparators. It also controls the output logic levels to the target. It is normally connected to Vdd I/O on the target board.
10	TDO	**JTAG Test Data Out** This input to the BDI3000 connects to the target TDO line.

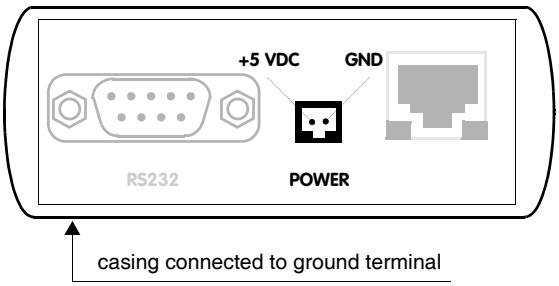
© Copyright 1997-2009 by ABATRON AG Switzerland

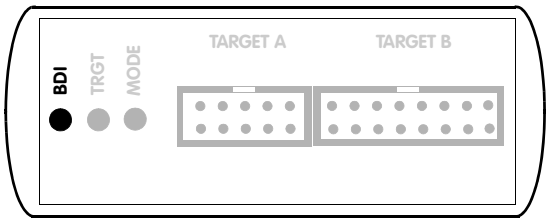
2.2 Connecting the BDI3000 to Power Supply

The BDI3000 needs to be supplied with the enclosed power supply from Abatron (5VDC).

Before use, check if the mains voltage is in accordance with the input voltage printed on power supply. Make sure that, while operating, the power supply is not covered up and not situated near a heater or in direct sun light. Dry location use only.

For error-free operation, the power supply to the BDI3000 must be between 4.75V and 5.25V DC. **The maximal tolerable supply voltage is 5.25 VDC. Any higher voltage or a wrong polarity might destroy the electronics.**





The green LED «BDI» marked light up when 5V power is connected to the BDI3000

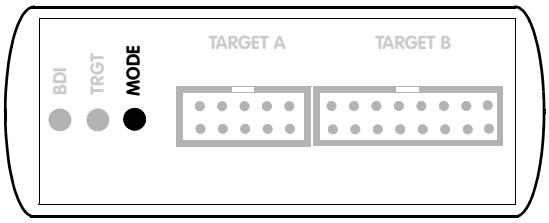
Please switch on the system in the following sequence:

- 1 –> external power supply
- 2 –> target system

© Copyright 1997-2009 by ABATRON AG Switzerland

2.3 Status LED «MODE»

The built in LED indicates the following BDI states:



MODE LED	BDI STATES
OFF	The BDI is ready for use, the firmware is already loaded.
ON	The output voltage from the power supply is too low.
BLINK	The BDI «loader mode» is active (an invalid firmware is loaded or loading firmware is active).

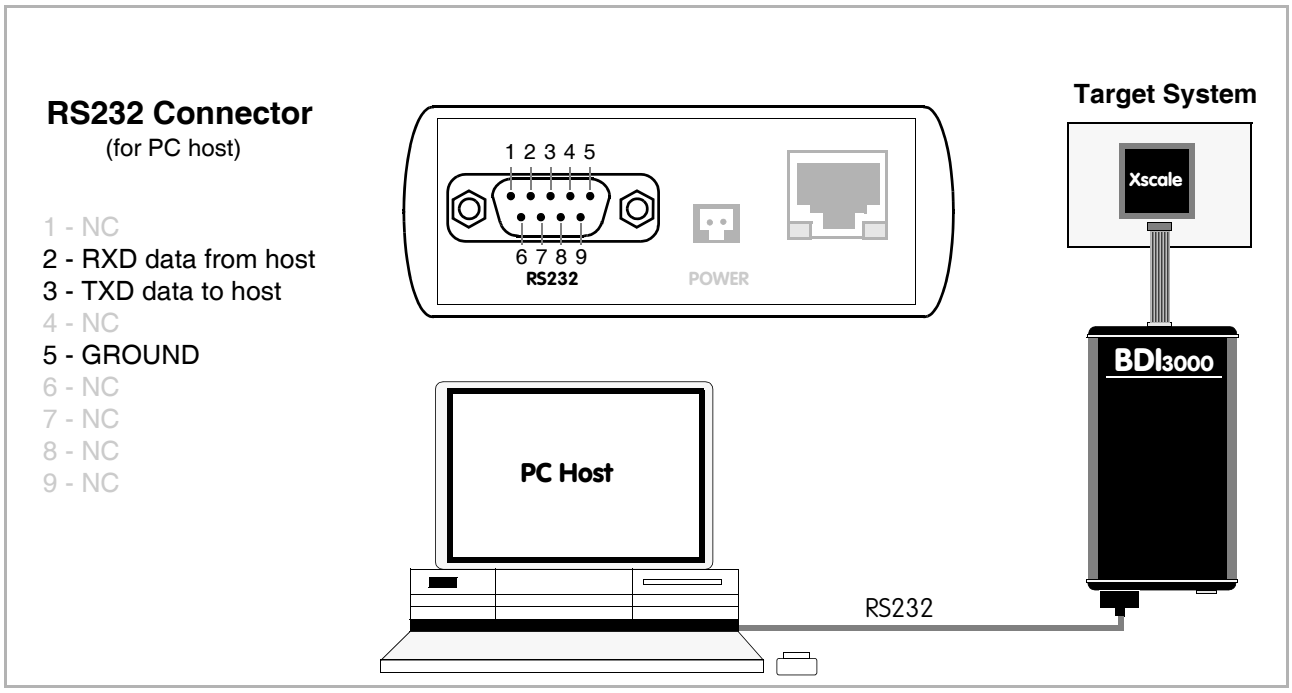
© Copyright 1997-2009 by ABATRON AG Switzerland

2.4 Connecting the BDI3000 to Host

2.4.1 Serial line communication

Serial line communication is only used for the initial configuration of the bdiGDB system.

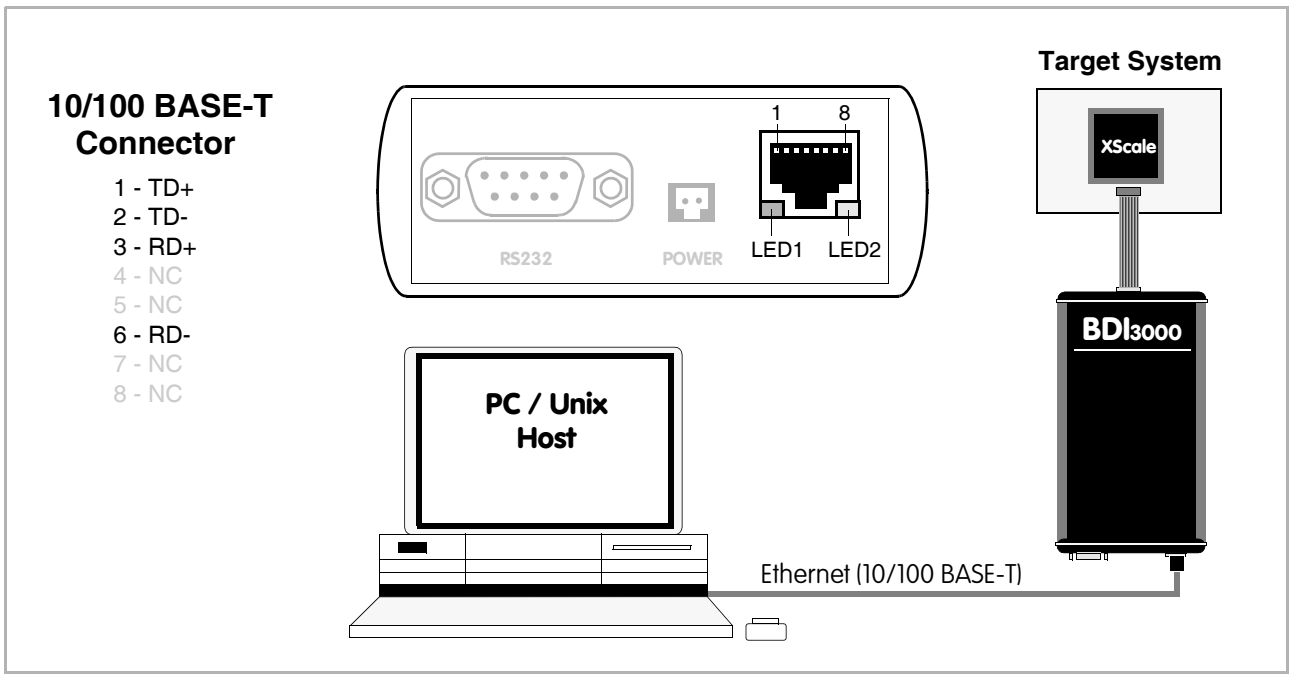
The host is connected to the BDI through the serial interface (COM1...COM4). The communication cable (included) between BDI and Host is a serial cable. There is the same connector pinout for the BDI and for the Host side (Refer to Figure below).



© Copyright 1997-2009 by ABATRON AG Switzerland

2.4.2 Ethernet communication

The BDI3000 has a built-in 10/100 BASE-T Ethernet interface (see figure below). Connect an UTP (Unshielded Twisted Pair) cable to the BD3000. Contact your network administrator if you have questions about the network.



The following explains the meanings of the built-in LED lights:

LED	Function	Description
LED 1 (green)	Link / Activity	When this LED light is ON, data link is successful between the UTP port of the BDI3000 and the hub to which it is connected. The LED blinks when the BDI3000 is receiving or transmitting data.
LED 2 (amber)	Speed	When this LED light is ON, 100Mb/s mode is selected (default). When this LED light is OFF, 10Mb/s mode is selected

© Copyright 1997-2009 by ABATRON AG Switzerland

2.5 Installation of the Configuration Software

On the enclosed diskette you will find the BDI configuration software and the firmware required for the BDI3000. For Windows users there is also a TFTP server included.

The following files are on the diskette.

b30xscgd.exe	Windows Configuration program
b30xscgd.xxx	Firmware for the BDI3000
tftpsrv.exe	TFTP server for Windows (WIN32 console application)
*.cfg	Configuration files
*.def	Register definition files
bdisetup.zip	ZIP Archive with the Setup Tool sources for Linux / UNIX hosts.

Overview of an installation / configuration process:

- Create a new directory on your hard disk
- Copy the entire contents of the enclosed diskette into this directory
- Linux only: extract the setup tool sources and build the setup tool
- Use the setup tool or Telnet (default IP) to load/update the BDI firmware
  Note: A new BDI has no firmware loaded.
- Use the setup tool or Telnet (default IP) to load the initial configuration parameters
  - IP address of the BDI.
  - IP address of the host with the configuration file.
  - Name of the configuration file. This file is accessed via TFTP.
  - Optional network parameters (subnet mask, default gateway).

Activating BOOTP:
The BDI can get the network configuration and the name of the configuration file also via BOOTP. For this simple enter 0.0.0.0 as the BDI's IP address (see following chapters). If present, the subnet mask and the default gateway (router) is taken from the BOOTP vendor-specific field as defined in RFC 1533.

With the Linux setup tool, simply use the default parameters for the -c option:

```
[root@LINUX_1 bdisetup]# ./bdisetup -c -p/dev/ttyS0 -b57
```

The MAC address is derived from the serial number as follows:
MAC: 00-0C-01-xx-xx-xx , replace the xx-xx-xx with the 6 left digits of the serial number
Example: SN# 33123407 ==>> 00-0C-01-33-12-34

Default IP: 192.168.53.72
Before the BDI is configured the first time, it has a default IP of 192.168.53.72 that allows an initial configuration via Ethernet (Telnet or Setup Tools). If your host is not able to connect to this default IP, then the initial configuration has to be done via the serial connection.

© Copyright 1997-2009 by ABATRON AG Switzerland

2.5.1 Configuration with a Linux / Unix host

The firmware update and the initial configuration of the BDI3000 is done with a command line utility. In the ZIP Archive bdisetup.zip are all sources to build this utility. More information about this utility can be found at the top in the bdisetup.c source file. There is also a make file included.
Starting the tool without any parameter displays information about the syntax and parameters.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000:

1. Build the setup tool:
The setup tool is delivered only as source files. This allows to build the tool on any Linux / Unix host. To build the tool, simply start the make utility.

```
[root@LINUX_1 bdisetup]# make
cc -O2   -c -o bdisetup.o bdisetup.c
cc -O2   -c -o bdicnf.o bdicnf.c
cc -O2   -c -o bdidll.o bdidll.c
cc -s bdisetup.o bdicnf.o bdidll.o -o bdisetup
```

2. Check the serial connection to the BDI:
With "bdisetup -v" you may check the serial connection to the BDI. The BDI will respond with information about the current loaded firmware and network configuration.
Note: Login as root, otherwise you probably have no access to the serial port.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : unknown
MAC      : ff-ff-ff-ff-ff-ff
IP Addr  : 255.255.255.255
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 255.255.255.255
Config   : ÿÿÿÿÿÿÿ........
```

3. Load/Update the BDI firmware:
With "bdisetup -u" the firmware is programmed into the BDI3000 flash memory. This configures the BDI for the target you are using. Based on the parameters -a and -t, the tool selects the correct firmware file. If the firmware file is in the same directory as the setup tool, there is no need to enter a -d parameter.

```
$ ./bdisetup -u -p/dev/ttyS0 -b115 -aGDB -tXSCALE
Connecting to BDI loader
Programming firmware with ./b30xscgd.100
Erasing firmware flash ....
Erasing firmware flash passed
Programming firmware flash ....
Programming firmware flash passed
```

© Copyright 1997-2009 by ABATRON AG Switzerland

4. Transmit the initial configuration parameters:
With "bdisetup -c" the configuration parameters are written to the flash memory within the BDI.
The following parameters are used to configure the BDI:

BDI IP Address	The IP address for the BDI3000. Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	The subnet mask of the network where the BDI is connected to. A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask. If the BDI and the host are in the same subnet, it is not necessary to enter a subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This file is read via TFTP. Keep in mind that TFTP has it's own root directory (usual /tftpboot). You can simply copy the configuration file to this directory and the use the file name without any path. For more information about TFTP use "man tftpd".

```
$ ./bdisetup -c -p/dev/ttyS0 -b115 \
> -i151.120.25.102 \
> -h151.120.25.112 \
> -fe:/bdi3000/mytarget.cfg
Connecting to BDI loader
Writing network configuration
Configuration passed
```

5. Check configuration and exit loader mode:
The BDI is in loader mode when there is no valid firmware loaded or you connect to it with the setup tool. While in loader mode, the Mode LED is blinking. The BDI will not respond to network requests while in loader mode. To exit loader mode, the "bdisetup -v -s" can be used. You may also power-off the BDI, wait some time (1min.) and power-on it again to exit loader mode.

```
$ ./bdisetup -v -p/dev/ttyS0 -b115 -s
BDI Type : BDI3000 (SN: 30000154)
Loader   : V1.00
Firmware : V1.00 bdiGDB for XScale
MAC      : 00-0c-01-30-00-01
IP Addr  : 151.120.25.102
Subnet   : 255.255.255.255
Gateway  : 255.255.255.255
Host IP  : 151.120.25.112
Config   : /bdi3000/mytarget.cfg
```

The Mode LED should go off, and you can try to connect to the BDI via Telnet.

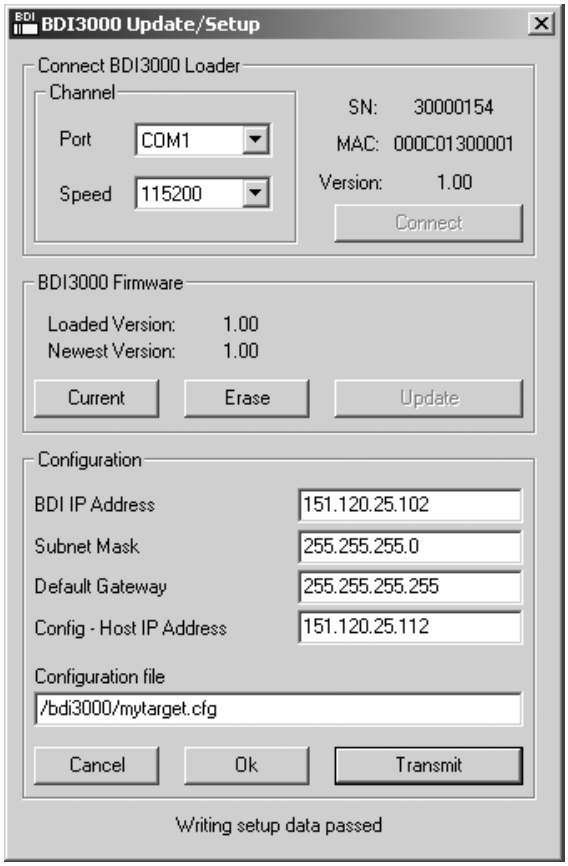
```
$ telnet 151.120.25.102
```

© Copyright 1997-2009 by ABATRON AG Switzerland

2.5.2 Configuration with a Windows host

First make sure that the BDI is properly connected (see Chapter 2.1 to 2.4).

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.



dialog box «BDI3000 Update/Setup»

Before you can use the BDI3000 together with the GNU debugger, you must store the initial configuration parameters in the BDI3000 flash memory. The following options allow you to do this:

Port	Select the communication port where the BDI3000 is connected during this setup session. If you select Network, make sure the Loader is already active (Mode LED blinking). If there is already a firmware loaded and running, use the Telnet command "boot loader" to activate Loader Mode.
Speed	Select the baudrate used to communicate with the BDI3000 loader during this setup session.
Connect	Click on this button to establish a connection with the BDI3000 loader. Once connected, the BDI3000 remains in loader mode until it is restarted or this dialog box is closed.
Current	Press this button to read back the current loaded BDI3000 firmware version. The current firmware version will be displayed.

© Copyright 1997-2009 by ABATRON AG Switzerland

Erase	Press this button to erase the current loaded firmware.
Update	This button is only active if there is a newer firmware version present in the execution directory of the bdiGDB setup software. Press this button to write the new firmware into the BDI3000 flash memory.
BDI IP Address	Enter the IP address for the BDI3000. Use the following format: xxx.xxx.xxx.xxx e.g.151.120.25.101 Ask your network administrator for assigning an IP address to this BDI3000. Every BDI3000 in your network needs a different IP address.
Subnet Mask	Enter the subnet mask of the network where the BDI is connected to. Use the following format: xxx.xxx.xxx.xxxe.g.255.255.255.0 A subnet mask of 255.255.255.255 disables the gateway feature. Ask your network administrator for the correct subnet mask.
Default Gateway	Enter the IP address of the default gateway. Ask your network administrator for the correct gateway IP address. If the gateway feature is disabled, you may enter 255.255.255.255 or any other value.
Config - Host IP Address	Enter the IP address of the host with the configuration file. The configuration file is automatically read by the BDI3000 after every start-up.
Configuration file	Enter the full path and name of the configuration file. This name is transmitted to the TFTP server when reading the configuration file.
Transmit	Click on this button to store the configuration in the BDI3000 flash memory.

Note:
Using this setup tool via the Network channel is only possible if the BDI3000 is already in Loader mode (Mode LED blinking). To force Loader mode, enter "boot loader" at the Telnet. The setup tool tries first to establish a connection to the Loader via the IP address present in the "BDI IP Address" entry field. If there is no connection established after a time-out, it tries to connect to the default IP (192.168.53.72).

© Copyright 1997-2009 by ABATRON AG Switzerland

2.5.3 Configuration via Telnet / TFTP

The firmware update and the initial configuration of the BDI3000 can also be done interactively via a Telnet connection and a running TFTP server on the host with the firmware file. In cases where it is not possible to connect to the default IP, the initial setup has to be done via a serial connection.

To avoid data line conflicts, the BDI3000 must be disconnected from the target system while programming the firmware for an other target CPU family.

Following the steps to bring-up a new BDI3000 or updating the firmware.
Connect to the BDI Loader via Telnet.
If a firmware is already running enter "boot loader" and reconnect via Telnet.

```
$ telnet 192.168.53.72
or
$ telnet <your BDI IP address>
```

Update the network parameters so it matches your needs:

```
LDR>network
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 192.168.53.72
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 255.255.255.255
    Config File :

LDR>netip 151.120.25.102
LDR>nethost 151.120.25.112
LDR>netfile /bdi3000/mytarget.cfg


LDR>network
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg

LDR>network save
saving network configuration ... passed
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg
```

In case the subnet has changed, reboot before trying to load the firmware

```
LDR>boot loader
```

© Copyright 1997-2009 by ABATRON AG Switzerland

Connect again via Telnet and program the firmware into the BDI flash:

```
$ telnet 151.120.25.102

LDR>info
    BDI Firmware: not loaded
    BDI CPLD ID : 01285043
    BDI CPLD UES: ffffffff
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg

LDR>fwload e:/temp/b30xscgd.100
erasing firmware flash ... passed
programming firmware flash ... passed

LDR>info
    BDI Firmware: 35 / 1.00
    BDI CPLD ID : 01285043
    BDI CPLD UES: ffffffff
    BDI MAC     : 00-0c-01-30-00-01
    BDI IP      : 151.120.25.102
    BDI Subnet  : 255.255.255.0
    BDI Gateway : 255.255.255.255
    Config IP   : 151.120.25.112
    Config File : /bdi3000/mytarget.cfg
LDR>
```

To boot now into the firmware use:

```
LDR>boot
```

The Mode LED should go off, and you can try to connect to the BDI again via Telnet.

```
telnet 151.120.25.102
```

© Copyright 1997-2009 by ABATRON AG Switzerland

2.6 Testing the BDI3000 to host connection

After the initial setup is done, you can test the communication between the host and the BDI3000. There is no need for a target configuration file and no TFTP server is needed on the host.

- If not already done, connect the BDI3000 system to the network.
- Power-up the BDI3000.
- Start a Telnet client on the host and connect to the BDI3000 (the IP address you entered during initial configuration).
- If everything is okay, a sign on message like «BDI Debugger for Embedded PowerPC» and a list of the available commands should be displayed in the Telnet window.

2.7 TFTP server for Windows

The bdiGDB system uses TFTP to access the configuration file and to load the application program. Because there is no TFTP server bundled with Windows, Abatron provides a TFTP server application **tftpsrv.exe**. This WIN32 console application runs as normal user application (not as a system service).

Command line syntax: tftpsrv [p] [w] [dRootDirectory]

Without any parameter, the server starts in read-only mode. This means, only read access request from the client are granted. This is the normal working mode. The bdiGDB system needs only read access to the configuration and program files.

The parameter [p] enables protocol output to the console window. Try it.
The parameter [w] enables write accesses to the host file system.
The parameter [d] allows to define a root directory.

`tftpsrv p`	Starts the TFTP server and enables protocol output
`tftpsrv p w`	Starts the TFTP server, enables protocol output and write accesses are allowed.
`tftpsrv dC:\tftp\`	Starts the TFTP server and allows only access to files in C:\tftp and its subdirectories. As file name, use relative names. For example "bdi\mpc750.cfg" accesses "C:\tftp\bdi\mpc750.cfg"

You may enter the TFTP server into the Startup group so the server is started every time you login.

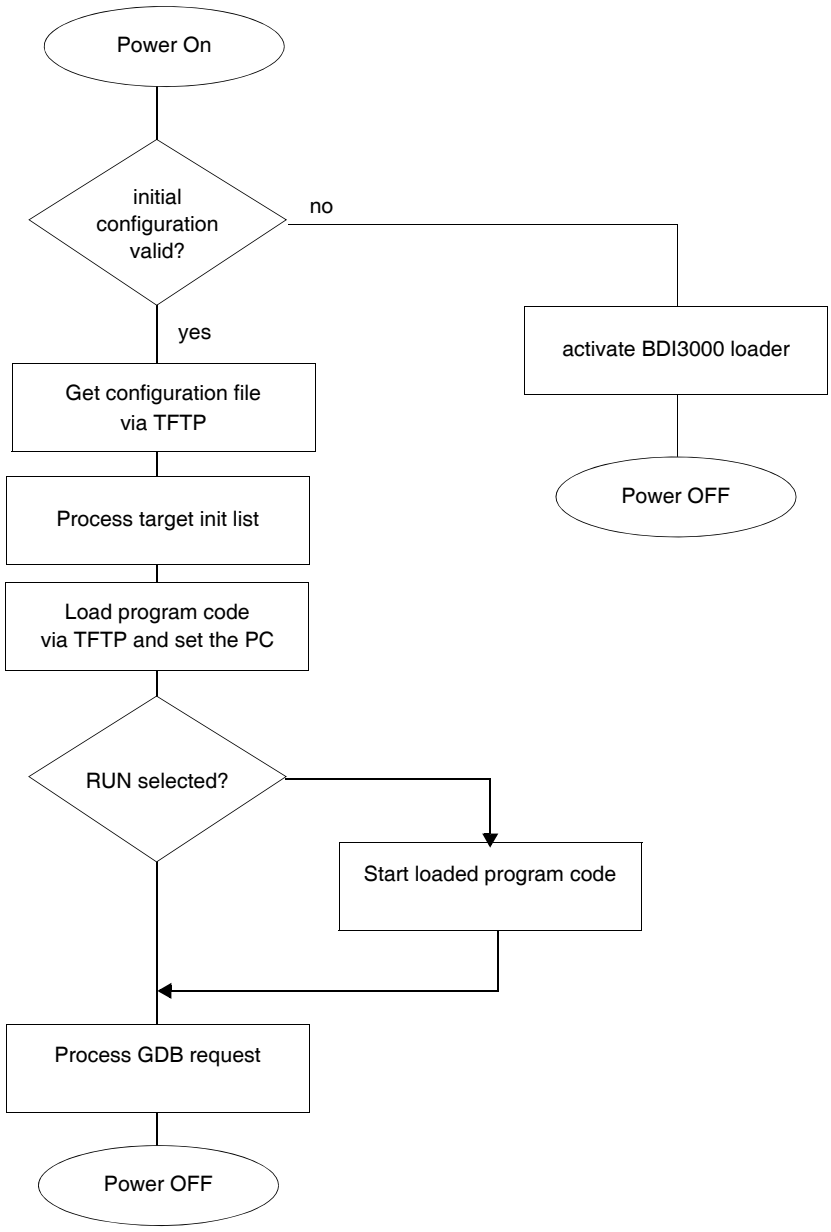
© Copyright 1997-2009 by ABATRON AG Switzerland

3 Using bdiGDB

3.1 Principle of operation

The firmware within the BDI handles the GDB request and accesses the target memory or registers via the JTAG interface. There is no need for any debug software on the target system. After loading the code via TFTP debugging can begin at the very first assembler statement.

Whenever the BDI system is powered-up the following sequence starts:



© Copyright 1997-2009 by ABATRON AG Switzerland

3.2 Configuration File

The configuration file is automatically read by the BDI3000 after every power on.
The syntax of this file is as follows:

```
; comment
[part name]
identifier parameter1 parameter2 ..... parameterN  ; comment
identifier parameter1 parameter2 ..... parameterN
.....
[part name]
identifier parameter1 parameter2 ..... parameterN
identifier parameter1 parameter2 ..... parameterN
.....
             etc.
```

Numeric parameters can be entered as decimal (e.g. 700) or as hexadecimal (0x80000).

© Copyright 1997-2009 by ABATRON AG Switzerland

3.2.1 Part [INIT]

The part [INIT] defines a list of commands which should be executed every time the target comes out of reset. The commands are used to get the target ready for loading the program file.

WGPR register value
Write value to the selected general purpose register.
register the register number 0 .. 15
value the value to write into the register
Example: WGPR 0 5

WREG name value
Write value to the selected CPU register by name
name the register name (CPSR)
value the value to write into the register
Example: WREG CPSR 0x600000D3

WCPn register value
Write value to the selected Coprocessor n register.
register the register number (see chapter CPxx Registers)
value the value to write into the register
Example: WCP15 0x010F 0x00002001 ;Enable CP0, CP13 access

WM8 address value
Write a byte (8bit) to the selected memory place.
address the memory address
value the value to write to the target memory
Example: WM8 0xFFFFFA21 0x04 ; SYPCR: watchdog disable ...

WM16 address value
Write a half word (16bit) to the selected memory place.
address the memory address
value the value to write to the target memory
Example: WM16 0x02200200 0x0002 ; TBSCR

WM32 address value
Write a word (32bit) to the selected memory place.
address the memory address
value the value to write to the target memory
Example: WM32 0x02200000 0x01632440 ; SIUMCR

FILL start size
Fill a memory range with its own address. Maybe used to initialize an ECC protected memory range. The start address has to be 32-bit aligned and the size has to be a multiple of 4.
start the start address of the memory range to fill
size the size in bytes of the memory range to fill
Example: FILL 0x00000000 0x10000000 ; fill 256MB ECC SDRAM

© Copyright 1997-2009 by ABATRON AG Switzerland

RM8 address value

Read a byte (8bit) from the selected memory place.

address the memory address

Example: RM8 0x00000000

RM16 address value

Read a half word (16bit) from the selected memory place.

address the memory address

Example: RM16 0x00000000

RM32 address value

Read a word (32bit) from the selected memory place.

address the memory address

Example: RM32 0x00000000

MMAP start end

Because a memory access to an invalid memory space via JTAG can lead to a deadlock, this entry can be used to define up to 32 valid memory ranges. If at least one memory range is defined, the BDI checks against this range(s) and avoids accessing of not mapped memory ranges.

start the start address of a valid memory range

end the end address of this memory range

Example: MMAP 0x00000000 0x03FFFFFF ;64MB Flash

DELAY value

Delay for the selected time.

value the delay time in milliseconds (1...30000)

Example: DELAY 500 ; delay for 0.5 seconds

© Copyright 1997-2009 by ABATRON AG Switzerland

3.2.2 Part [TARGET]

The part [TARGET] defines some target specific values.

CPUTYPE type

This value gives the BDI information about the connected CPU.

type: 80200, IOP310, IOP321, IOP331, IOP332
PXA210, PXA250, PXA270, PXA300, PXA310, PXA320
IXP420, IXP421, IXP422, IXP425, IXP460, IXP465
IXP430, IXP431, IXP432, IXP433, IXP435
IXP2300, IXP2325, IXP2350, IXP2400, IXP2800
MANZANO, MONAHANS

Example: CPUTYPE PXA270

JTAGCLOCK value

With this value you can select the JTAG clock rate the BDI3000 uses when communication with the target CPU.

value:

0 = 32 MHz	7 = 500 kHz
1 = 16 MHz	8 = 200 kHz
2 = 11 MHz	9 = 100 kHz
3 = 8 MHz	10 = 50 kHz
4 = 5 MHz	11 = 20 kHz
5 = 4 MHz	12 = 10 kHz
6 = 1 MHz	

Example: CLOCK 1 ; JTAG clock is 16 MHz

RESET time

This parameter allows to define a time (in ms) the BDI asserts the reset signal. By default the reset signal is asserted for at least 500 ms.

time: The time in milliseconds the BDI assert the reset signal.

Example: RESET 1000 ; assert RESET for 1 second

TRST type

This parameter allows to define the TRST diver type. By default a push-pull diver is used. If necessary an open-drain driver can be selected.

type: PUSHPULL (default)
OPENDRAIN

Example: TRST OPENDRAIN ; use open-drain driver for TRST

POWERUP delay [RESET]

The value entered in this configuration line is the delay time in milliseconds the BDI waits before it begins the reset sequence. This time should be longer than the on-board reset circuit asserts RESET (default is 2 seconds). If the RESET option is also present, the BDI asserts the reset signal via the debug connector as soon as power-up is detected.

delay: the power-up start delay in milliseconds

Example: POWERUP 5000 ;start delay after power-up

WAKEUP time

This entry in the init list allows to define a delay time (in ms) the BDI inserts between releasing the reset line and starting communicating with the target.

time: the delay time in milliseconds

Example: WAKEUP 3000 ; insert 3sec wake-up time

© Copyright 1997-2009 by ABATRON AG Switzerland

DBGHANDLER addr — This parameter defines the base address of the debug handler. The debug handler code (and the override vector tables) are loaded into the mini instruction cache during reset processing. See also chapter "Debug Handler" below and XScale core manual chapter "Software Debug".
Note: This parameter is not used for Manzano and Monahans processors.

value	2k aligned in the range 0x00000000 ... 0x01FEF800 or 0xFE000800 ... 0xFFFFF800
Example:	DBGHANDLER 0x00000000

STARTUP mode [runtime] — This parameter selects the target startup mode. The following modes are supported:

RESET	This default mode forces the target to debug mode immediately out of reset. No code is executed after reset.
STOP	In this mode, the BDI lets the target execute code for "runtime" milliseconds after reset. This mode is useful when monitor code should initialize the target system.
RUN	After reset, the target executes code until stopped by the Telnet "halt" command.
Example:	STARTUP STOP 3000 ; let the CPU run for 3 seconds

BDIMODE mode param — This parameter selects the BDI debugging mode. The following modes are supported:

LOADONLY	Loads and starts the application code. No debugging via JTAG interface.
AGENT	The debug agent runs within the BDI. There is no need for any debug software on the target. This mode accepts a second parameter. If RUN is entered as a second parameter, the loaded application will be started immediately, otherwise only the PC is set and BDI waits for GDB requests.
Example:	BDIMODE AGENT RUN

VECTOR CATCH mask — When this line is present, the BDI traps exceptions. The mask is used to set the Trap bits in the Debug Control rRegister (DCSR).

mask	selects the exceptions to catch
Example:	VECTOR CATCH 0x1E ;catch Abort, SWI, Undef

BREAKMODE mode — This parameter defines how breakpoints are implemented.

SOFT	This is the normal mode. Breakpoints are implemented by replacing code with a BKPT instruction.
HARD	In this mode, the XScale breakpoint hardware is used. Only 2 breakpoints at a time is supported.
Example:	BREAKMODE HARD

© Copyright 1997-2009 by ABATRON AG Switzerland

ENDIAN format

This entry defines the endiannes of the memory system.

format	The endiannes of the target memory: LITTLE (default), BIG
Example:	ENDIAN LITTLE

VTABLO [vn] opcode

When this entry is present without the optional vector number (vn), the BDI fills the default vector table in the Mini ICache (except the debug/reset vector) with the requested opcode. Also the vector table in the Mini ICache will not be updated each time before the target is restarted. When the optional vector number (vn) is used, individual opcodes can be assigned to the individual vectors. See also chapter 3.3.5 Debug Handler.
Note: This parameter is not used for Manzano and Monahans processors.

vn	The vector number 0 .. 7
opcode	The opcode to load into the default vector table.
Example:	VTABLO 0xe59ff018 ;Fixed vectors "ldr pc, [pc, #18]"

VTABHI [vn] opcode

The same as VTABLO but for the relocated vector table at 0xFFFF0000.
Note: This parameter is not used for Manzano and Monahans processors.

vn	The vector number 0 .. 7
opcode	The opcode to load into the relocated vector table.
Example:	VTABHI 0xffffffff ;Do not update relocated vector table

SIO port [baudrate]

When this line is present, a TCP/IP channel is routed to the BDI's RS232 connector. The port parameter defines the TCP port used for this BDI to host communication. You may choose any port except 0 and the default Telnet port (23). On the host, open a Telnet session using this port. Now you should see the UART output in this Telnet session. You can use the normal Telnet connection to the BDI in parallel, they work completely independent. Also input to the UART is implemented.

port	The TCP/IP port used for the host communication.
baudrate	The BDI supports 2400 ... 115200 baud
Example:	SIO 7 9600 ;TCP port for virtual IO

© Copyright 1997-2009 by ABATRON AG Switzerland

Daisy chained JTAG devices:
For XScale targets, the BDI can also handle systems with multiple devices connected to the JTAG scan chain. In order to put the other devices into BYPASS mode and to count for the additional bypass registers, the BDI needs some information about the scan chain layout. Enter the number (count) and total instruction register (irlen) length of the devices present before the XScale chip (Predecessor). Enter the appropriate information also for the devices following the XScale chip (Successor):

SCANPRED count irlen — This value gives the BDI information about JTAG devices present before the XScale chip in the JTAG scan chain.

count	The number of preceding devices (0 ... 31)
irlen	The sum of the length of all preceding instruction registers (IR) (0 ... 1024)
Example:	SCANPRED 1 8 ; one device with an IR length of 8

SCANSUCC count irlen — This value gives the BDI information about JTAG devices present after the XScale chip in the JTAG scan chain.

count	The number of succeeding devices (0 ... 31)
irlen	The sum of the length of all succeeding instruction registers (IR) (0 ... 1024)
Example:	SCANSUCC 2 12 ; two device with an IR length of 8+4

© Copyright 1997-2009 by ABATRON AG Switzerland

3.2.3 Part [HOST]

The part [HOST] defines some host specific values.

IP ipaddress

The IP address of the host.

ipaddress	the IP address in the form xxx.xxx.xxx.xxx
Example:	IP 151.120.25.100

FILE filename

The default name of the file that is loaded into RAM using the Telnet 'load' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name.

filename	the filename including the full path or $ for relative path.
Example:	FILE F:\gnu\demo\xscale\test.elf FILE $test.elf

FORMAT format [offset]

The format of the image file and an optional load address offset. If the image is already stored in ROM on the target, select ROM as the format. The optional parameter "offset" is added to any load address read from the image file.

format	SREC, BIN, AOUT, ELF, COFF or ROM
Example:	FORMAT ELF FORMAT ELF 0x10000

LOAD mode

In Agent mode, this parameters defines if the code is loaded automatically after every reset.

mode	AUTO, MANUAL
Example:	LOAD MANUAL

START address

The address where to start the program file. If this value is not defined and the core is not in ROM, the address is taken from the code file. If this value is not defined and the core is already in ROM, the PC will not be set before starting the target. This means, the program starts at the normal reset address (0x00000000).

address	the address where to start the program file
Example:	START 0x10000

DEBUGPORT port [RECONNECT]

The TCP port GDB uses to access the target. If the RECONNECT parameter is present, an open TCP/IP connection (Telnet/GDB) will be closed if there is a connect request from the same host (same IP address).

port	the TCP port number (default = 2001)
Example:	DEBUGPORT 2001

© Copyright 1997-2009 by ABATRON AG Switzerland

PROMPT string

This entry defines a new Telnet prompt. The current prompt can also be changed via the Telnet interface.

Example: PROMPT IQ80310>

DUMP filename

The default file name used for the Telnet DUMP command.

filename the filename including the full path

Example: DUMP dump.bin

TELNET mode

By default the BDI sends echoes for the received characters and supports command history and line editing. If it should not send echoes and let the Telnet client in "line mode", add this entry to the configuration file.

mode ECHO (default), NOECHO or LINE

Example: TELNET NOECHO ; use old line mode

© Copyright 1997-2009 by ABATRON AG Switzerland

3.2.4 Part [FLASH]

The Telnet interface supports programming and erasing of flash memories. The bdiGDB system has to know which type of flash is used, how the chip(s) are connected to the CPU and which sectors to erase in case the ERASE command is entered without any parameter.

CHIPTYPE type

This parameter defines the type of flash used. It is used to select the correct programming algorithm.

format AM29F, AM29BX8, AM29BX16, I28BX8, I28BX16, AT49, AT49X8, AT49X16, STRATAX8, STRATAX16, MIRROR, MIRRORX8, MIRRORX16, S29M64X8, S29M32X16, M58X32, AM29DX16, AM29DX32

Example: CHIPTYPE AT49X16

CHIPSIZE size

The size of **one** flash chip in bytes (e.g. AM29F010 = 0x20000). This value is used to calculate the starting address of the current flash memory bank.

size the size of one flash chip in bytes

Example: CHIPSIZE 0x80000

BUSWIDTH width

Enter the width of the memory bus that leads to the flash chips. Do not enter the width of the flash chip itself. The parameter CHIPTYPE carries the information about the number of data lines connected to one flash chip. For example, enter 16 if you are using two AM29F010 to build a 16bit flash memory bank.

with the width of the flash memory bus in bits (8 | 16 | 32)

Example: BUSWIDTH 16

FILE filename

The default name of the file that is programmed into flash using the Telnet 'prog' command. This name is used to access the file via TFTP. If the filename starts with a $, this $ is replace with the path of the configuration file name. This name may be overridden interactively at the Telnet interface.

filename the filename including the full path or $ for relative path.

Example: FILE F:\gnu\xscale\bootrom.hex
FILE $bootrom.hex

FORMAT format [offset]

The format of the file and an optional address offset. The optional parameter "offset" is added to any load address read from the program file.

format SREC, BIN, AOUT, ELF or COFF

Example: FORMAT SREC
FORMAT ELF 0x10000

WORKSPACE address

If a workspace is defined, the BDI uses a faster programming algorithm that runs out of RAM on the target system. Otherwise, the algorithm is processed within the BDI. The workspace is used for a 1kByte data buffer and to store the algorithm code. There must be at least 2kBytes of RAM available for this purpose.

address the address of the RAM area

Example: WORKSPACE 0x00000000

© Copyright 1997-2009 by ABATRON AG Switzerland

ERASE addr [increment count] [mode [wait]]

The flash memory may be individually erased or unlocked via the Telnet interface. In order to make erasing of multiple flash sectors easier, you can enter an erase list. All entries in the erase list will be processed if you enter ERASE at the Telnet prompt without any parameter. This list is also used if you enter UNLOCK at the Telnet without any parameters. With the "increment" and "count" option you can erase multiple equal sized sectors with one entry in the erase list.

address	Address of the flash sector, block or chip to erase
increment	If present, the address offset to the next flash sector
count	If present, the number of equal sized sectors to erase
mode	BLOCK, CHIP, UNLOCK Without this optional parameter, the BDI executes a sector erase. If supported by the chip, you can also specify a block or chip erase. If UNLOCK is defined, this entry is also part of the unlock list. This unlock list is processed if the Telnet UNLOCK command is entered without any parameters. **Note:** Chip erase does not work for large chips because the BDI time-outs after 3 minutes. Use block erase.
wait	The wait time in ms is only used for the unlock mode. After starting the flash unlock, the BDI waits until it processes the next entry.
Example:	ERASE 0xff040000 ;erase sector 4 of flash ERASE 0xff060000 ;erase sector 6 of flash ERASE 0xff000000 CHIP ;erase whole chip(s) ERASE 0xff010000 UNLOCK 100 ;unlock, wait 100ms ERASE 0xff000000 0x10000 7 ; erase 7 sectors

Example for the IQ80310 board (I28F640J3A):

```
[FLASH]
WORKSPACE   0xa0020000  ;workspace in target RAM for fast programming algorithm
CHIPTYPE    STRATAX8    ;Flash type
CHIPSIZE    0x800000    ;The size of one flash chip in bytes
BUSWIDTH    8           ;The width of the flash memory bus in bits (8 | 16 | 32)
FILE          E:\temp\dump512k.bin
FORMAT      BIN 0x00080000
ERASE       0x00080000  ;erase sector 4
ERASE       0x000A0000  ;erase sector 5
ERASE       0x000C0000  ;erase sector 6
ERASE       0x000E0000  ;erase sector 7
```

the above erase list maybe replaces with:

```
ERASE         0x00080000 0x20000 4 ;erase 4 sectors
```

© Copyright 1997-2009 by ABATRON AG Switzerland

Supported Flash Memories:
There are currently 3 standard flash algorithm supported. The AMD, Intel and Atmel AT49 algorithm. Almost all currently available flash memories can be programmed with one of this algorithm. The flash type selects the appropriate algorithm and gives additional information about the used flash.

For 8bit only flash:	AM29F (MIRROR), I28BX8, AT49
For 8/16 bit flash in 8bit mode:	AM29BX8 (MIRRORX8), I28BX8 (STRATAX8), AT49X8
For 8/16 bit flash in 16bit mode:	AM29BX16 (MIRRORX16), I28BX16 (STRATAX16), AT49X16
For 16bit only flash:	AM29BX16, I28BX16, AT49X16
For 16/32 bit flash in 16bit mode:	AM29DX16
For 16/32 bit flash in 32bit mode:	AM29DX32
For 32bit only flash:	M58X32

Some newer Spansion MirrorBit flashes cannot be programmed with the MIRRORX16 algorithm because of the used unlock address offset. Use S29M32X16 for these flashes.

The AMD and AT49 algorithm are almost the same. The only difference is, that the AT49 algorithm does not check for the AMD status bit 5 (Exceeded Timing Limits).
Only the AMD and AT49 algorithm support chip erase. Block erase is only supported with the AT49 algorithm. If the algorithm does not support the selected mode, sector erase is performed. If the chip does not support the selected mode, erasing will fail. The erase command sequence is different only in the 6th write cycle. Depending on the selected mode, the following data is written in this cycle (see also flash data sheets): 0x10 for chip erase, 0x30 for sector erase, 0x50 for block erase.
To speed up programming of Intel Strata Flash and AMD MirrorBit Flash, an additional algorithm is implemented that makes use of the write buffer. This algorithm needs a workspace, otherwise the standard Intel/AMD algorithm is used.

The following table shows some examples:

Flash	x 8	x 16	x 32	Chipsize
Am29F010	AM29F	-	-	0x020000
Am29F800B	AM29BX8	AM29BX16	-	0x100000
Am29DL323C	AM29BX8	AM29BX16	-	0x400000
Am29PDL128G	-	AM29DX16	AM29DX32	0x01000000
Intel 28F032B3	I28BX8	-	-	0x400000
Intel 28F640J3A	STRATAX8	STRATAX16	-	0x800000
Intel 28F320C3	-	I28BX16	-	0x400000
AT49BV040	AT49	-	-	0x080000
AT49BV1614	AT49X8	AT49X16	-	0x200000
M58BW016BT	-	-	M58X32	0x200000
SST39VF160	-	AT49X16	-	0x200000
Am29LV320M	MIRRORX8	MIRRORX16	-	0x400000

© Copyright 1997-2009 by ABATRON AG Switzerland

Note:
Some Intel flash chips (e.g. 28F800C3, 28F160C3, 28F320C3) power-up with all blocks in locked state. In order to erase/program those flash chips, use the init list to unlock the appropriate blocks:

```
WM16    0xFFF00000    0x0060        unlock block 0
WM16    0xFFF00000    0x00D0
WM16    0xFFF10000    0x0060        unlock block 1
WM16    0xFFF10000    0x00D0
        ....
WM16    0xFFF00000    0xFFFF        select read mode
```

or use the Telnet "unlock" command:

```
UNLOCK [<addr> [<delay>]]
```

addr	This is the address of the sector (block) to unlock
delay	A delay time in milliseconds the BDI waits after sending the unlock command to the flash. For example, clearing all lock-bits of an Intel J3 Strata flash takes up to 0.7 seconds.

If "unlock" is used without any parameter, all sectors in the erase list with the UNLOCK option are processed.

To clear all lock-bits of an Intel J3 Strata flash use for example:

```
BDI> unlock 0xFF000000 1000
```

To erase or unlock multiple, continuous flash sectors (blocks) of the same size, the following Telnet commands can be used:

```
ERASE  <addr> <step> <count>
UNLOCK <addr> <step> <count>
```

addr	This is the address of the first sector to erase or unlock.
step	This value is added to the last used address in order to get to the next sector. In other words, this is the size of one sector in bytes.
count	The number of sectors to erase or unlock.

The following example unlocks all 256 sectors of an Intel Strata flash (28F256K3) that is mapped to 0x00000000. In case there are two flash chips to get a 32bit system, double the "step" parameter.

```
BDI> unlock 0x00000000 0x20000 256
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.2.5 Part [REGS]

In order to make it easier to access target registers via the Telnet interface, the BDI can read in a register definition file. In this file, the user defines a name for the register and how the BDI should access it (e.g. as memory mapped, memory mapped with offset, ...). The name of the register definition file and information for different registers type has to be defined in the configuration file.
The register name, type, address/offset/number and size are defined in a separate register definition file. This way, you can create one register definition file for a specific target processor that can be used for all possible positions of the internal memory map. You only have to change one entry in the configuration file.

An entry in the register definition file has the following syntax:

```
name   type   addr   size
```

name	The name of the register (max. 12 characters)
type	The register type

GPR	General purpose register
CP15	Coprocessor 15 register
CP14	Coprocessor 14register
....	
CP0	Coprocessor 0 register
MM	Absolute direct memory mapped register
DMM1...DMM4	Relative direct memory mapped register
IMM1...IMM4	Indirect memory mapped register

addr	The address, offset or number of the register
size	The size (8, 16, 32) of the register, default is 32

The following entries are supported in the [REGS] part of the configuration file:

FILE filename

The name of the register definition file. This name is used to access the file via TFTP. The file is loaded once during BDI startup.

filename	the filename including the full path
Example:	FILE C:\bdi\regs\reg40400.def

DMMn base

This defines the base address of direct memory mapped registers. This base address is added to the individual offset of the register.

base	the base address
Example:	DMM1 0x01000

IMMn addr data

This defines the addresses of the memory mapped address and data registers of indirect memory mapped registers. The address of a IMMn register is first written to "addr" and then the register value is access using "data" as address.

addr	the address of the Address register
data	the address of the Data register
Example:	DMM1 0x04700000

© Copyright 1997-2009 by ABATRON AG Switzerland

Example for a register definition (IQ80310):

Entry in the configuration file:

```
[REGS]
FILE            E:\cygwin\home\bdidemo\xscale\reg80310.def
```

The register definition file:

```
;
;Coprocessor Register Numbers for xScale:
;
;       +-----+-+-------+-----+-+-------+
;       |opc_2|0|  CRm  |opc_1|0|  nbr  |
;       +-----+-+-------+-----+-+-------+
;
;The 16bit register number is used to build the appropriate MCR/MRC instruction.
;
;
;name         type          addr          size
;-------------------------------------------
;
id            CP15          0x0000        32      ;ID code
cache         CP15          0x2000        32      ;Cache type
ctr           CP15          0x0001        32      ;Control
aux           CP15          0x2001        32      ;Auxiliary Control
ttb           CP15          0x0002        32      ;Translation Table Base
dac           CP15          0x0003        32      ;Domain Access Control
fsr           CP15          0x0005        32      ;Fault Status
far           CP15          0x0006        32      ;Fault Address
pid           CP15          0x000d        32      ;Process ID
cpacc         CP15          0x010f       32      ;Coprocessor Access

;
;
;       80312 Memory controller
;
sdir          MM            0x00001500
sdcr          MM            0x00001504
sdbr          MM            0x00001508
sbr0          MM            0x0000150c
sbr1          MM            0x00001510
;
eccr          MM            0x00001534
elog0         MM            0x00001538
elog1         MM            0x0000153c
ecar0         MM            0x00001540
ecar1         MM            0x00001544
ectst         MM            0x00001548
febr0         MM            0x0000154c
febr1         MM            0x00001550
fbsr0         MM            0x00001554
fbsr1         MM            0x00001558
fwsr0         MM            0x0000155c
fwsr1         MM            0x00001560
mcisr         MM            0x00001564
rfr           MM            0x00001568
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.3 Debugging with GDB

Because the target agent runs within BDI, no debug support has to be linked to your application. There is also no need for any BDI specific changes in the application sources. Your application must be fully linked because no dynamic loading is supported.

3.3.1 Target setup

Target initialization may be done at two places. First with the BDI configuration file, second within the application. The setup in the configuration file must at least enable access to the target memory where the application will be loaded. Disable the watchdog and setting the CPU clock rate may also be done with the BDI configuration file. Application specific initializations like setting the timer rate are best located in the application startup sequence.

3.3.2 Connecting to the target

As soon as the target comes out of reset, BDI initializes it and loads your application code. If RUN is selected, the application is immediately started, otherwise only the target PC is set. BDI now waits for GDB request from the debugger running on the host.

After starting the debugger, it must be connected to the remote target. This can be done with the following command at the GDB prompt:

```
(gdb)target remote bdi3000:2001
```

bdi3000	This stands for an IP address. The HOST file must have an appropriate entry. You may also use an IP address in the form xxx.xxx.xxx.xxx
2001	This is the TCP port used to communicate with the BDI

If not already suspended, this stops the execution of application code and the target CPU changes to background debug mode.

Remember, every time the application is suspended, the target CPU is freezed. During this time no hardware interrupts will be processed.

Note: For convenience, the GDB detach command triggers a target reset sequence in the BDI.

```
(gdb)...
(gdb)detach
... Wait until BDI has resetet the target and reloaded the image
(gdb)target remote bdi3000:2001
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.3.3 Breakpoint Handling

There are two breakpoint modes supported. One of them (SOFT) is implemented by replacing application code with a BKPT instruction. The other (HARD) uses the built in breakpoint logic. If HARD is selected, only up to 2 breakpoints can be active at the same time.

The following example selects SOFT as the breakpoint mode:

```
BREAKMODE   SOFT      ;SOFT or HARD, HARD uses hardware breakpoints
```

The BDI supports only a GDB version that uses a Z-Packet to set breakpoints (GDB Version 5.0 or newer). GDB tells the BDI to set / clear breakpoints with this special protocol unit. The BDI will respond to this request by replacing code in memory with the BKPT instruction or by setting the appropriate hardware breakpoint.

3.3.4 GDB monitor command

The BDI supports the GDB V5.x "monitor" command. Telnet commands are executed and the Telnet output is returned to GDB.

```
(gdb) target remote bdi3000:2001
Remote debugging using bdi3000:2001
0x10b2 in start ()
(gdb) monitor md 0 1
00000000 : 0xe59ff018  - 442503144  ...
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.3.5 Debug handler

Note: This chapter does not apply to Manzano and Monahans based processors. The XScale core used in this processors uses special Debug SRAM for the handler and the vectors are not overridden by a Mini ICache entry.

The XScale variant of debugging via JTAG depends on a debug handler running on the target. This handler communicates with the BDI via the JTAG interface. This debug handler is loaded into the mini instruction cache via JTAG during reset processing. Please read also the chapter "Software Debug" in the XScale manual.
It is also necessary that the reset vectors at 0x00000000 and 0xFFFF0000 are overridden. Because it is not possible to override only the reset vector, all vectors are overridden by a valid entry in the Mini ICache. A code fetch always accesses the vector table in the Mini ICache loaded via JTAG. The Mini ICache is never updated from memory, it can only be loaded via JTAG.
Always before the target exits debug mode, the BDI reads back the vector tables from memory and updates the Mini ICache accordingly. This works fine when the vector table is only updated while the target is in debug mode (e.g. via application download). If the application updates the vector table on the fly, this will not change the Mini ICache and the old vectors will still be used until the target enters/exits debug mode at least once.
Therefore it is recommended that the vector table itself is not change dynamically. A second table outside the Mini ICache addresses may be used to change exception handler addresses on the fly.

In order to force an enter/exit debug mode sequence, a "bkpt #1" instruction maybe added after the vector tables are updated. The BDI recognizes this "bkpt #1" instruction and immediately restarts the target. If you cannot change the code, set a hardware breakpoint via Telnet that triggers after the vector table has been updated by the code running on the target.

For Linux you may use a data breakpoint ("bd w 0xffff0200"). This breakpoint triggers in the "__trap_init()" function after the vector table has been written. After the breakpoint has triggered, delete it and restart execution. The vector table in the Mini IC is now updated.

Alternatively, if you know the values that will be assigned to the vector table(s) you can force the BDI to fill the Mini ICache with this values. In this case there is no need to force the target into debug mode after "__trap_init()" has setup the vector table at 0xFFFF0000. The configuration then may looks as follows:

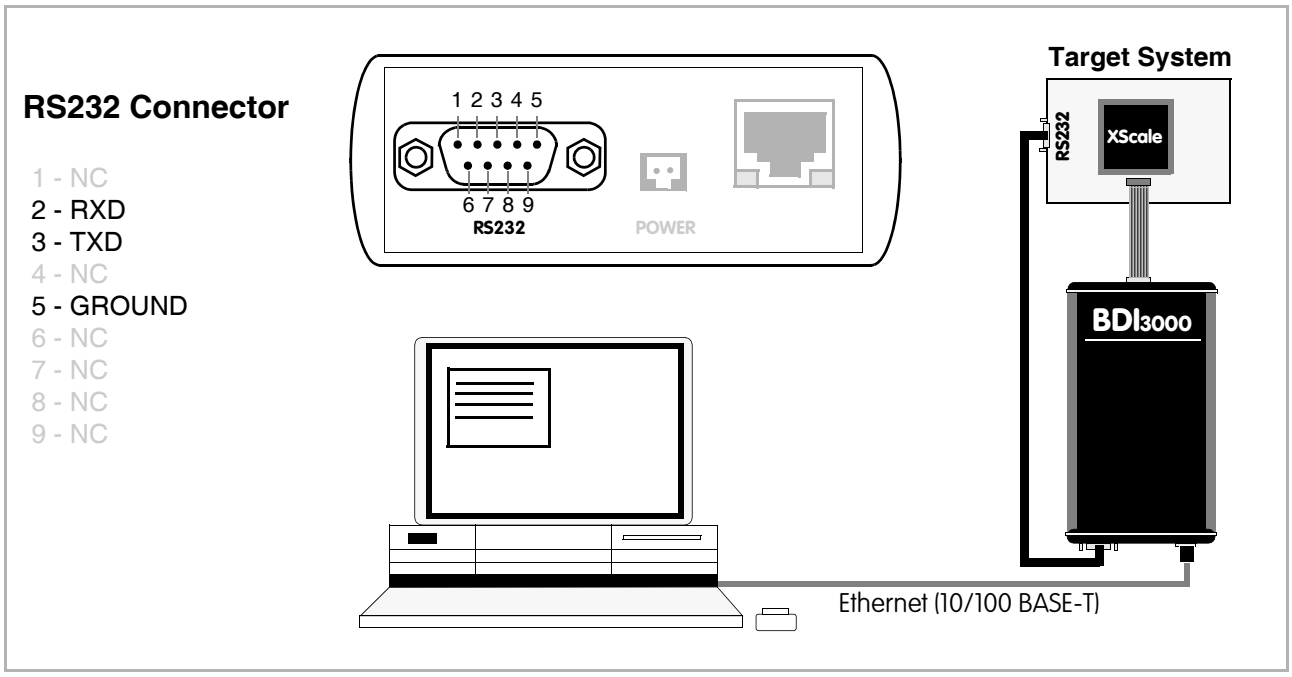
```
VTABLO    0xe59ff018    ;Use fixed vectors "ldr pc, [pc, #18]" (RedBoot)
VTABHI 0  0xef9f0000    ;Write to vector 0 forces the BDI to use fixed vectors.
VTABHI 1  0xea0000dd    ;0xffff0004:     b       0xffff0380
VTABHI 2  0xe59ff410    ;0xffff0008:     ldr     pc, [pc, #410]  ; 0xffff0420
VTABHI 3  0xea0000bb    ;0xffff000c:     b       0xffff0300
VTABHI 4  0xea00009a    ;0xffff0010:     b       0xffff0280
VTABHI 5  0xea0000fa    ;0xffff0014:     b       0xffff0404
VTABHI 6  0xea000078    ;0xffff0018:     b       0xffff0200
VTABHI 7  0xea0000f7    ;0xffff001c:     b       0xffff0400
```

Important:
The necessary allocation of the vector tables in the Mini ICache is the reason for many troubles when debugging a XScale target via a JTAG tool. Therefore it is important that you exactly understand the involved mechanism. The vector table in the Mini ICache has to match the one in memory at least at the time when an interrupt occurs. Otherwise the system will crash. The easiest way is to use a static vector table (e.g. "ldr pc, [pc, #18]"). You can still dynamically update the address of the interrupt handler via the address table that is use to load the PC. Force the BDI to load the correct opcodes into the Mini IC via the configuration file and all vector table troubles are gone.

© Copyright 1997-2009 by ABATRON AG Switzerland

3.3.6 Target serial I/O via BDI

A RS232 port of the target can be connected to the RS232 port of the BDI3000. This way it is possible to access the target's serial I/O via a TCP/IP channel. For example, you can connect a Telnet session to the appropriate BDI3000 port. Connecting GDB to a GDB server (stub) running on the target should also be possible.



The configuration parameter "SIO" is used to enable this serial I/O routing.
The used framing parameters are 8 data, 1 stop and not parity.

```
[TARGET]
....
SIO    7      9600          ;Enable SIO via TCP port 7 at 9600 baud
```

Warning!!!
Once SIO is enabled, connecting with the setup tool to update the firmware will fail. In this case either disable SIO first or disconnect the BDI from the LAN while updating the firmware.

© Copyright 1997-2009 by ABATRON AG Switzerland

3.4 Telnet Interface

A Telnet server is integrated within the BDI. The Telnet channel is used by the BDI to output error messages and other information. Also some basic debug tasks may be done by using this interface. Enter help at the Telnet command prompt to get a list of the available commands.

Telnet Debug features:

- Display and modify memory locations
- Display and modify registers
- Single step a code sequence
- Set hardware breakpoints (for code and data accesses)
- Load a code file from any host
- Start / Stop program execution
- Programming and Erasing Flash memory

During debugging with GDB, the Telnet is mainly used to reboot the target (generate a hardware reset and reload the application code). It may be also useful during the first installation of the bdiGDB system or in case of special debug needs.

Multiple commands separated by a semicolon can be entered on one line.

Example of a short Telnet session:

```
CSB226>rd
GPR00: 00000040 00000008 00000807 00000000
GPR04: 48000040 43f3ffdd 48000040 ffff0db0
GPR08: 58ff186c f99deaf1 afba5b18 fd747be3
GPR12: 7b75faee fd7f7f3f ffff0b28 00000060
PC   : 00000060    CPSR: 000000d3
CSB226>ti
    Core number       : 0
    Core state        : debug mode (ARM)
    Debug entry cause : Single Step
    Current PC        : 0x00000064
    Current CPSR      : 0x000000d3 (Supervisor)
CSB226>md 0
00000000 : ea000016 e59ff014 e59ff014 e59ff014  ................
00000010 : e59ff014 e59ff014 e59ff014 e59ff014  ................
00000020 : a1fe0160 a1fe01c0 a1fe0220 a1fe0280  `.....  .......
00000030 : a1fe02e0 a1fe0340 a1fe03a0 deadbeef  ....@...........
00000040 : a1fe0000 a1fe0000 a1ff57fc a1ff6c90  .........W...l..
00000050 : a1ff57fc a1ff6c90 0badc0de a1fe0000  .W...l..........
```

Notes:
The DUMP command uses TFTP to write a binary image to a host file. Writing via TFTP on a Linux/ Unix system is only possible if the file already exists and has public write access. Use "man tftpd" to get more information about the TFTP server on your host.

© Copyright 1997-2009 by ABATRON AG Switzerland

3.4.1 Command list

```
"MD    [<address>] [<count>]  display target memory as word (32bit)",
"MDH   [<address>] [<count>]  display target memory as half word (16bit)",
"MDB   [<address>] [<count>]  display target memory as byte (8bit)",
"DUMP  <addr> <size> [<file>] dump target memory to a file",
"MM    <addr> <value> [<cnt>] modify word(s) (32bit) in target memory",
"MMH   <addr> <value> [<cnt>] modify half word(s) (16bit) in target memory",
"MMB   <addr> <value> [<cnt>] modify byte(s) (8bit) in target memory",
"MT    <addr> <count>         memory test",
"MC    [<address>] [<count>]  calculates a checksum over a memory range",
"MV                           verifies the last calculated checksum",
"FILL  [<address>] [<count>]  fill a memory range with its own address",
"RD    [<name>]               display general purpose or user defined register",
"RDUMP [<file>]               dump all user defined register to a file",
"RDALL                        display all ARM registers ",
"RDCP  [<cp>] <number>        display CP register, default is CP15",
"RDACC [<number>]             display an internal accumulator",
"RM    {<nbr>|<name>} <value> modify general purpose or user defined register",
"RMCP  [<cp>] <number><value> modify CP register, default is CP15",
"RMACC [<nbr>] <high> <low>   modify an internal accumulator",
"DCACHE <addr | set>          display L1 data cache (only Manzano)",
"ICACHE <addr | set>          display L1 instruction cache (only Manzano)",
"L2TAG  <from> [<to>]         display L2 cache state (only Manzano)",
"RESET [HALT | RUN [time]]    reset the target system, change startup mode",
"GO    [<pc>]                 set PC and start target system",
"GO    <n> <n> [<n>[<n>]]     start multiple cores in requested order",
"TI    [<pc>]                 single step an instruction",
"HALT  [<n>[<n>[<n>[<n>]]]]   force core(s) to debug mode (n = core number)",
"BI    <addr>                 set instruction breakpoint",
"CI    [<id>]                 clear instruction breakpoint(s)",
"BD    [R|W] <addr> [<mask>]  set data breakpoint",
"CD    [<id>]                 clear data watchpoint(s)",
"TRAP  <mask>                 set the Vector Trap Bits (0x80=TF,..,0x01=TR)",
"INFO                         display information about the current state",
"LOAD   [<offset>] [<file> [<format>]] load program file to target memory",
"VERIFY [<offset>] [<file> [<format>]] verify a program file to target memory",
"PROG   [<offset>] [<file> [<format>]] program flash memory",
"                           <format>  : SREC, BIN, AOUT, ELF or COFF",
"ERASE  [<address> [<mode>]]  erase a flash memory sector, chip or block",
"                  <mode>  : CHIP, BLOCK or SECTOR (default is sector)",
"ERASE  <addr> <step> <count> erase multiple flash sectors",
"UNLOCK [<addr> [<delay>]]    unlock a flash sector",
"UNLOCK <addr> <step> <count> unlock multiple flash sectors",
"FLASH  <type> <size> <bus>   change flash configuration",
"DELAY  <ms>                  delay for a number of milliseconds",
"SELECT <core>                change the current core",
"HOST   <ip>                  change IP address of program file host",
"PROMPT <string>              defines a new prompt string",
"CONFIG                       display or update BDI configuration",
"CONFIG <file> [<hostIP> [<bdiIP> [<gateway> [<mask>]]]]",
"HELP                         display command list",
"BOOT  [loader]               reboot the BDI and reload the configuration",
"QUIT                         terminate the Telnet session"
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.4.2 CPxx Registers

Via Telnet it is possible to access the Coprocessor 15,14,13 registers. Following the Telnet commands that are used to access CP registers:

```
"RDCP    <number>              display control processor 15 register",
"RDCP 15 <number>              display control processor 15 register",
"RDCP 14 <number>              display control processor 14 register",
"RDCP 13 <number>              display control processor 13 register",
             ....

"RMCP    <number>   <value>    modify control processor 15 register",
"RMCP 15 <number>   <value>    modify control processor 15 register",
"RMCP 14 <number>   <value>    modify control processor 14 register",
"RMCP 13 <number>   <value>    modify control processor 13 register",
             ....
```

The parameter number selects the CPxx register. This parameter is used to build the appropriate MCR or MRC instruction.

```
+-----+-+-------+-----+-+-------+
|opc_2|0|  CRm  |opc_1|0|  nbr  |
+-----+-+-------+-----+-+-------+
```

Some examples:

CP15 : ID register (CRn = 0, opcode_2 = 0)

```
BDI> rdcp 15 0x0000
```

CP15 : Cache Type (CRn = 0, opcode_2 = 1)

```
BDI> rdcp 15 0x2000
```

CP15 : Invalidate I cache line (CRn = 7, opcode_2 = 1, CRm = 5)

```
BDI> rmcp 15 0x2507  0xA0000000
```

© Copyright 1997-2009 by ABATRON AG Switzerland

3.5 Multi-Core Support

The bdiGDB system supports concurrent debugging of up to 4 XScale cores connected to the same JTAG scan chain. For every core you can start its own GDB session. The default port numbers used to attach the remote targets are 2001 ... 2004. In the Telnet you switch between the cores with the command "select <0..3>". In the configuration file, simply begin the line with the appropriate core number. If there is no #n in front of a line, the BDI assumes core #0.

The following example defines two PXA250 cores on the scan chain.

```
[TARGET]
JTAGCLOCK       1               ;use 8 MHz JTAG clock
POWERUP         2000            ;start delay after power-up detected in ms
WAKEUP          1000            ;give reset time to complete

#0 CPUTYPE      PXA250          ;the target CPU type
#0 SCANPRED     0 0             ;JTAG devices connected before this core
#0 SCANSUCC     1 5             ;JTAG devices connected after this core
#0 DBGHANDLER   0xFFFF0800      ;debug handler base address
#0 ENDIAN       LITTLE          ;memory model (LITTLE | BIG)
#0 BREAKMODE    SOFT            ;SOFT or HARD
#0 VECTOR       CATCH 0xDE      ;trap all vectors
#0 VTABHI       0xffffffff      ;Do not update relocated vector table

#1 CPUTYPE      PXA250          ;the target CPU type
#1 SCANPRED     1 5             ;JTAG devices connected before this core
#1 SCANSUCC     0 0             ;JTAG devices connected after this core
#1 DBGHANDLER   0xFFFF0800      ;debug handler base address
#1 ENDIAN       LITTLE          ;memory model (LITTLE | BIG)
#1 BREAKMODE    SOFT            ;SOFT or HARD
#1 VECTOR       CATCH 0xDE      ;trap all vectors
#1 VTABLO       0xEA000012      ;Use fixed vectors "b xxx"
#1 VTABHI       0xffffffff      ;Do not update relocated vector table
```

For a complete configuration example see "old226_new226.cfg" on the diskette. This configuration was used to debug two daisy chained Cogent CSB226 PXA250 evaluation boards.

© Copyright 1997-2009 by ABATRON AG Switzerland

4 Specifications

Operating Voltage Limiting	5 VDC ± 0.25 V
Power Supply Current	typ. 500 mA max. 1000 mA
RS232 Interface: Baud Rates	9'600,19'200, 38'400, 57'600,115'200
Data Bits	8
Parity Bits	none
Stop Bits	1
Network Interface	10/100 BASE-T
BDM/JTAG clock	up to 32 MHz
Supported target voltage	1.2 – 5.0 V
Operating Temperature	+ 5 °C ... +60 °C
Storage Temperature	-20 °C ... +65 °C
Relative Humidity (noncondensing)	<90 %rF
Size	160 x 85 x 35 mm
Weight (without cables)	280 g
Host Cable length (RS232)	2.5 m
Electromagnetic Compatibility	CE compliant
Restriction of Hazardous Substances	RoHS 2002/95/EC compliant

Specifications subject to change without notice

© Copyright 1997-2009 by ABATRON AG Switzerland

5 Environmental notice

Disposal of the equipment must be carried out at a designated disposal site.

6 Declaration of Conformity (CE)

CE

DECLARATION OF CONFORMITY

This declaration is valid for following product:

Type of device: BDM/JTAG Interface
Product name: BDI3000

The signing authorities state, that the above mentioned equipment meets the requirements for emission and immunity according to

EMC Directive 89/336/EEC

The evaluation procedure of conformity was assured according to the following standards:

IEC 61000-6-2: 1999, mod. EN61000-6-2: 2001
IEC 61000-6-3: 1996, mod. EN61000-6-2: 2001

This declaration of conformity is based on the test report no. E1087-05-7a of Quinel, Zug, Swiss Testing Service, accreditation no. STS 037

Manufacturer:

ABATRON AG
Lettenstrasse 9
CH-6343 Rotkreuz

Authority:

Max Vock
Marketing Director

Ruedi Dummermuth
Technical Director

Rotkreuz, 7/18/2007

© Copyright 1997-2009 by ABATRON AG Switzerland

7 Warranty

ABATRON Switzerland warrants the physical CD, cable and BDI3000 to be free of defects in materials and workmanship for a period of 3 years following the date of purchase when used under normal conditions.

In the event of notification within the warranty period of defects in material or workmanship, ABATRON will replace/repair defective CD, cable or BDI3000. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited loss of profit, special, incidental, consequential, or other similar claims.
ABATRON Switzerland specifically disclaims all other warranties - expressed or implied, including but not limited to implied warranties of merchantability and fitness for particular purposes - with respect to defects in the CD, cable and BDI3000, and the program license granted herein, including without limitation the operation of the program with respect to any particular application, use, or purposes. In no event shall ABATRON be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential, or other damages.

Failure in handling which leads to defects are not covered under this warranty. The warranty is void under any self-made repair operation.

© Copyright 1997-2009 by ABATRON AG Switzerland

Appendices

A Troubleshooting

Problem
The firmware can not be loaded.

Possible reasons

- The BDI is not correctly connected with the Host (see chapter 2).
- A wrong communication port is selected (Com 1...Com 4).
- The BDI is not powered up

Problem
No working with the target system (loading firmware is okay).

Possible reasons

- Wrong pin assignment (BDM/JTAG connector) of the target system (see chapter 2).
- Target system initialization is not correctly –> enter an appropriate target initialization list.
- An incorrect IP address was entered (BDI3000 configuration)
- BDM/JTAG signals from the target system are not correctly (short-circuit, break, ...).
- The target system is damaged.

Problem
Network processes do not function (loading the firmware was successful)

Possible reasons

- The BDI3000 is not connected or not correctly connected to the network (LAN cable or media converter)
- An incorrect IP address was entered (BDI3000 configuration)

© Copyright 1997-2009 by ABATRON AG Switzerland

B Maintenance

The BDI needs no special maintenance. Clean the housing with a mild detergent only. Solvents such as gasoline may damage it.

C Trademarks

All trademarks are property of their respective holders.

© Copyright 1997-2009 by ABATRON AG Switzerland